

NATIONAL PARK SERVICE

Recommended Database Strategies

including I&M Database Template

Version 1.0

Prepared by NPS/AKSO/GIS/Angela Southwould

DRAFT COPY

Recommended Database Strategies

*Creating a database can be like creating a universe,
only more complicated.
At least when the universe was created,
there was no one around to complain.
-Michael J. Hernandez*

Overview	2
Relational Databases	3
Tables	3
Fields	5
Relationships	6
Relational Database Management Systems (RDBMS)	7
Data Normalization	9
Spatial Data – ArcView Shape Files	15
The Database Template	16
Study Table	17
Site Table	20
Observation Table	25
How to Implement the Database Template and Naming Standards	28
Adapting to the National Database Template	35
Guidelines to Database Design	38
Table Names	39
How To Name a Table	39
Field Names	41
How to Name a Field	41
Define a Field Validation Rule	48
Define a Field Label	49
Define a Field Format/Input Mask	49
Other Data Design Topics	51
Actual Measurement vs. Ranges and Categories	51
Data Verification	52
References	53

Overview

This document will provide you with some guidelines for creating a standardized database. I will touch on the topic of relational databases; but keep in mind this subject is very robust and cannot be fully described in a few pages of a single document. I include an overview of the I&M Database Template. This covers the core tables and instructions for implementing the template into your project. The remainder of the document focuses on a foundation of standards related to naming conventions. These rules are easily implemented into new and existing databases. These standards also fit with the template. The last section of the document contains some commentary on miscellaneous topics that are sometimes the source of problems and confusion during the design process.

Recommended Database Strategies

Relational Databases

A database is a collection of data that is organized for a particular purpose. Flat file databases store all data in a single table, usually in a grid-like format. Relational databases are a better choice, because they make data viewing and manipulation easier by eliminating unnecessary duplication of data. Unlike flat files, which store all data in a single table, relational databases separate data by subject into different tables that are then related by common fields. The primary goal of good relational database is to make sure that your data can be easily maintained over time.

Benefits of a Relational Database

- ✓ Eliminates duplicate information
- ✓ Eases data entry and maintenance
- ✓ Eases data viewing and querying

Tables

A **table** is the primary structure in a relational database. A table should always describe a single subject. The subject may be an object, such as an observer, a site, or a tree found at a site. The subject of a table may also represent an event, or something that occurs at a specific point in time, such as an observation. A table contains fields and records. A **field**, or attribute, represents a characteristic of the table subject. A **record** represents a single instance of the table subject and contains each field, whether or not that field contains a value. Each table should contain a single field, or combination of fields that makes up the **primary key**. The primary key uniquely identifies a record within the table. A single table should never contain duplicate fields or redundant data.

Example:

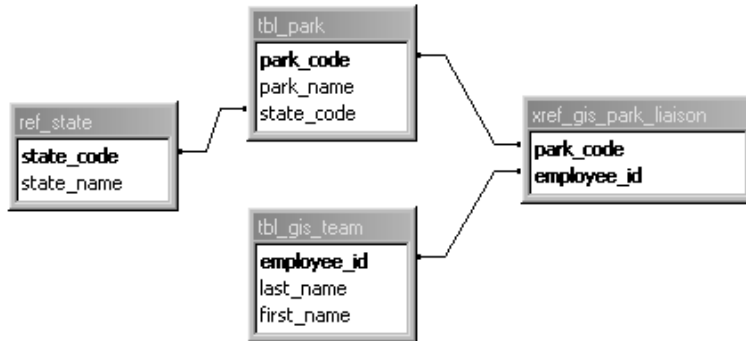
Check #	Date	Pay To	Amount
2001	10-Jan	Moose's Tooth Pizza	\$15.00
2002	10-Jan	GCI Long Distance	\$41.28
2003	15-Jan	Chugach Electric	\$31.02
2004	17-Jan	Alaska Bagel Factory	\$4.00
2005	19-Jan	Barnes and Noble	\$12.50
2006	20-Jan	Moose's Tooth Pizza	\$22.00

Here is a simple example of a table. The subject is an object, your checkbook registry. Each column in the registry is a field. The fields of data are: check number, check date, check payee, and check amount. Each row in the registry represents a check; or a record in your table. The primary key is the check number, because this value will locate one and only one check in your registry. Regardless of the number of checks you write, you will never use the same check number twice.

A table may fall into one of three primary categories. A **data table** supplies information and is the most common table type. A **reference table** stores a list of valid values for a field referenced in another data table. The final table type is more complex. A **cross-reference table** indicates a many-to-many relationship between two or more tables and links those tables together. The table category will come into play later, when you are creating and naming your tables.

Recommended Database Strategies

Example:



This entity-relationship diagram shows four tables. The park and team tables are both data tables. The park table contains a field for the state abbreviation. Since there are a limited, known number of states, we can define this list and store the values in a reference table for states. We can build the database such that for the state field in the park table, a user may only enter a value if it is already stored in the state table. On a data entry window, this would usually show up in the form of a drop down list, or pick list. The gis park liaison is a cross-reference table. Our model requires this table because each park can have multiple liaisons and each gis team member can be a liaison for more than one park. This is a many-to-many relationship and is accurately represented by creating a new table to fit between and cross-reference the two data tables.

Important Elements of a Table

- ✓ Table Name
- ✓ Table Category {Data, Reference, Cross-Reference}
- ✓ Table Description

Primary Pitfalls to Avoid in Tables

- 💣 Tables that contain data about more than one subject
- 💣 Tables that do not have a primary key

Recommended Database Strategies

Fields

A **field**, as described above, is a characteristic of the subject of the table. Each field should contain a single value that cannot be broken down into smaller components. This means choosing to store a user's first name, middle initial, and last name separately, rather than all together in a single field for the full name. Another example of this is separating the city, state, and zip code into three unique fields. This allows for easier implementation of validation rules and simplifies querying of the data. Each field should be unique within the table structure; this means avoiding repetitive fields such as tree1, tree2, and tree3 within a single record. If your model allows multiple values like this for a single subject, then a related table is a better design choice (see the following section on relationships). A field that is used in more than one table should maintain all of it's elements, including the field name.

Important Elements of a Field

General Elements

- ✓ Field Name
- ✓ Table Name
- ✓ Field Description

Table Elements

- ✓ Type of Key {None, Primary, Foreign, Alternate}
- ✓ Unique Value {Yes, No}
- ✓ Required Value {Yes, No}
- ✓ Field Data Type {Boolean, Alphanumeric, Numeric, DateTime, Counter}
 - ✓ Valid Characters {Letters, Numbers, Special Characters}
 - ✓ Length, Decimal Places

Data Entry Elements

- ✓ Field Label
- ✓ Values Entered By {User, System}
- ✓ Input Mask
- ✓ Display Format
- ✓ Default Value

Validation/Domain Elements

- ✓ Range of Allowed Values
- ✓ Reference Table Containing Valid Values

Query Elements

- ✓ Comparisons Allowed {=, !=, >, >=, <, <=}
- ✓ Operations Allowed {Addition, Subtraction, Multiplication, Division}

Primary Pitfalls to Avoid in Fields

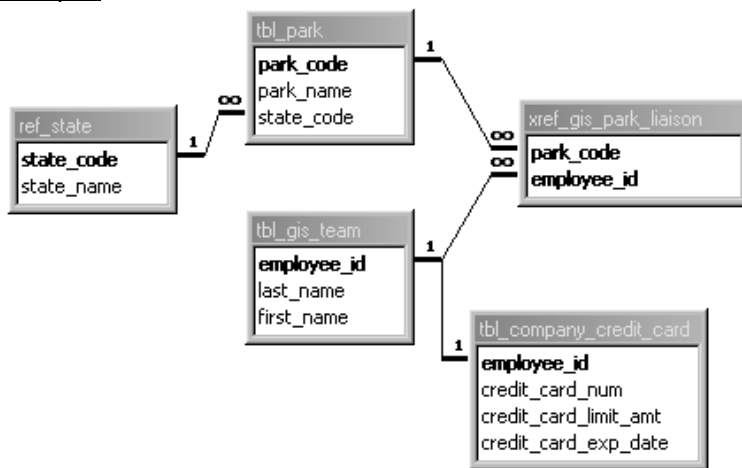
- 🚫 Fields containing multiple values
- 🚫 Fields containing calculated values
- 🚫 Repeating data fields

Recommended Database Strategies

Relationships

A **relationship** is a connection between two tables. Relationships are important to reduce duplicate or redundant data. This improves the integrity of your data, and improves the ease of data entry and querying. There are three types of relationships: **one-to-one**, **one-to-many**, or **many-to-many**. In a one-to-one (1:1) relationship, one record in the first table is related to zero or one record in the second table and vice versa. In a one-to-many (1:M) relationship, one record in the first table is related to zero or more records in the second table and a record in the second table is related back to only one record in the first table. In a many-to-many (M:M) relationship, one record in the first table is related to zero or more records in the second table. And one record in the second table is related to zero or more records in the first table. This special case is implemented by inserting a third table, the cross-reference table, between the two tables that represent the M:M relationship.

Example:



This entity-relationship diagram shows four tables that are related to one another. Each park may reside in only a single state (an assumption for this example). Each state may accommodate any number of parks. There is a 1:M relationship between states and parks. As described in the previous example, each park may have multiple liaisons and each gis team member may be a liaison in many parks. This defines a M:M relationship. We must create a third table, the cross-reference table, to accurately implement the M:M relationship and link these tables. Now, the model actually displays a 1:M relationship between parks and liaisons and a 1:M relationship between gis team members and liaisons, which is the same as a M:M relationship between parks and gis team members. The 1:1 relationship is less common. It is represented here by the relationship between an employee and a company credit card. Each employee (gis team member) may have zero or one credit card. Each company credit card belongs to one and only one employee.

Important Elements of a Relationship

- ✓ Parent Table Name
- ✓ Child Table Name
- ✓ Relationship Type {1:1, 1:M, M:M}

Primary Pitfall to Avoid in Relationships

- ☛ Not using relationships; or creating fields in a single table that should be broken into two related tables

This is a very brief overview of relationships, since they are such a complex topic. I include this just to familiarize you with the terms and different types of relationships.

Recommended Database Strategies

Relational Database Management Systems (RDBMS)

Microsoft Access is the NPS database standard. You are **strongly** encouraged to store your tabular data in related tables within Access, or at least another relational database management system such as SQL Server. Most real world data is relational and only an RDBMS allows relationships to be accurately depicted. Relationships eliminate duplicate and redundant data; this improves your data integrity. Each value that is unnecessarily duplicated in your database increases your maintenance effort and the overall risk of error.

Non-relational data storage methods are called **flat files**. Forcing relational data into a flat file is poor practice. In a flat file, all data is stored in a single grid-like structure. Text files, Excel spreadsheets, and dBase DBFs are examples of flat files.

Example:

In this spreadsheet (flat file), there are only two different observation sites, but the latitude, longitude, and elevation are repeated each time the site is observed. This opens your data up for error. For site 2301102, the data entry person may mistakenly enter 7000 as the elevation for this site in row 8 when the value should be 700. If you are viewing rows 4 through 9, it may be apparent that a data entry error was made. But if the spreadsheet is ordered differently and you are only viewing row 8, the error may not jump out at you. Another problem can occur if later, the observer realizes that the elevation for site 2301102 should be 1700, not 700. All rows for the site must be modified. Again, if the spreadsheet is not sorted by site number, the user may not update all occurrences of that site. If only half of the rows are caught, then half of the rows for site 2301102 read an elevation of 700 and the other half read 1700. When the errors are finally realized at a later date, someone must conduct research to find the correct value.

	A	B	C	D	E	F	G	H
1	site_num	latitude_deg	longitude_deg	elevation_ft	observer_name	observation_date	tree_species_code	tree_height_m
2	1850201	62.2554	-135.56671	2000	ALS	2002-01-20	PICMAR	6
3	1850201	62.2554	-135.56671	2000	ALS	2002-01-20	BETPAR	20
4	2301102	69.32677	-125.69411	700	ALS	1999-07-05	BETPAR	2
5	2301102	69.32677	-125.69411	700	ALS	1999-07-05	PICGLA	4
6	2301102	69.32677	-125.69411	700	ALS	1999-07-05	PICMAR	5
7	2301102	69.32677	-125.69411	700	ALS	2002-01-21	BETPAR	3
8	2301102	69.32677	-125.69411	700	ALS	2002-01-21	PICGLA	6
9	2301102	69.32677	-125.69411	700	ALS	2002-01-21	PICMAR	6

Rather than using a flat file, such as a spreadsheet, this data is better represented in a relational model, as seen below. A relational model splits this spreadsheet into three tables so that the problems described above will not occur. These three tables are linked together to accurately depict the same data. The site table contains only information about the site: latitude, longitude, and elevation. These values occur once. If an error is made and the data needs updated, only one change is necessary. The observation table contains information about the observation: date and observer. The site number is included as a linking field so that you know which observation goes with which site. The tree table contains the information unique to the trees observed: tree species and tree height. Again there are linking fields. The site number and observation date tie the tree data back to the other two tables.

One nice thing about relational data: even though the data is split among three separate tables, you can still view the data in a grid style format for reporting purposes using a query, if necessary.

Recommended Database Strategies

Example: (continued)



The diagram above shows how the tables are linked together. One site can have multiple observations over time. During each observation, the user can record multiple trees at the site. These are 1:M relationships.

The tables to the right have been color-coded to show how the related data fits within the three tables. The dark green record in the site table is related to the other highlighted records. The yellow record in the observation table is related to the yellow records in the tree table. The same is true for the bright green records. This example demonstrates 1-to-many relationships. One site may have many observations. One observation may have detected many trees.

tbl_site : Table			
site_num	latitude_deg	longitude_deg	elevation_ft
1850201	62.2554	-135.56671	2000
2301102	69.32677	-125.69411	700
*			
Record: 1 of 2			

tbl_observation : Table		
site_num	observation_date	observer_name
1850201	1/20/2002	ALS
2301102	7/5/1999	ALS
2301102	1/21/2002	ALS
*		
Record: 1 of 3		

tbl_tree : Table			
site_num	observation_date	tree_species	tree_height_m
1850201	1/20/2002	BETPAR	20
1850201	1/20/2002	PICMAR	6
2301102	7/5/1999	BETPAR	2
2301102	7/5/1999	PICGLA	4
2301102	7/5/1999	PICMAR	5
2301102	1/21/2002	BETPAR	3
2301102	1/21/2002	PICGLA	6
2301102	1/21/2002	PICMAR	6
*			
Record: 1 of 8			

Relational Data “Red Flag”

Finding 1-to-many or many-to-many relationships in your data should signify to you that a relational database is the best choice for your data!

In this case, a flat file will not accurately model your data.

Recommended Database Strategies

Data Normalization

The process of converting data from a flat file format into a relational database is **normalization**. Normalization separates the fields from a large table into multiple, smaller, related tables by removing all unnecessary or duplicate fields, eliminating redundant data, and ensuring that each table represents only one subject. In doing this, data viewing and manipulation can be accomplished in the easiest manner.

Data normalization can be a difficult concept for spreadsheet experts. One good reason for using a relational database over a flat file is that it allows your data to grow without causing problems. A common occurrence in spreadsheets is the addition of new columns and/or worksheets to accommodate new data. For a site stored in a single spreadsheet row, you might create several columns for the trees that have been observed at each site (i.e. tree1, tree_height1, tree2, tree_height2...tree10, tree_height10). If you come across a site that has 11 different tree species, you add the new columns tree11 and tree_height11. You might also add a new worksheet for each year's worth of subsequent observations. This can cause lots of copying and pasting or rewriting of formulas and macros. Additionally, the data can be difficult to view, validate, and analyze.

When databases are designed properly and normalized, they can accommodate new data (such as any number of tree observations) without modification to the table structure and previously built queries and reports. Once this concept is understood, you can build your database to take advantage of this.

This process can be broken down into six steps of normalization. Our aim is to complete the first three steps with your database, getting into third normal form. The remaining forms are not necessary for our purposes, as they can sometimes "over-normalize" a table to the point where the value of normalization is lost and the table becomes difficult to use. Normalization can best be described with an example, which follows the descriptions of each individual normal form.

Example:

Here is a spreadsheet (flat file) in its original layout. This table is not normalized.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	Site Observation Data 1999-2002														
2	site_num	park_code	latitude_deg	longitude_deg	elevation_ft	observer_name	observer_dept	observer_phone_num	observation_date	tree1_species_code	tree1_height_m	tree2_species_code	tree2_height_m	tree3_species_code	tree3_height_m
3	2301102	YUCH	69.32677	-125.69411	700	LEM	BIO	9075552032	1999-07-05	BETPAR	2	PICGLA	4	PICMAR	5
4	1850201	YUCH	62.2554	-135.56671	2000	DEL	BIO	9075552033	2000-06-15	PICMAR	6	BETPAR	19	PICGLA	2
5	1850201	YUCH	62.2554	-135.56671	2000	ALS	GIS	9075550001	2002-01-20	PICMAR	6	BETPAR	20		
6	2301102	YUCH	69.32677	-125.69411	700	ALS	GIS	9075550001	2002-01-21	BETPAR	3	PICGLA	6	PICMAR	6

Recommended Database Strategies

Signs of a Non-Normalized Table

- ☛ All data is stored in a single table
- ☛ A single record contains information on more than one subject
- ☛ A single record contains repeating fields
- ☛ Multiple records contain the same values for a group of fields

Step 1: First Normal Form – Eliminate repeating groups

The first step in normalizing your data is to get it into **first normal form**. Each set of related attributes should be created in a separate table. Each table should contain a set of attributes that form a unique key. So far, this follows the primary table rules that each table must contain attributes for a single subject and each table must contain an attribute (or set of attributes) that uniquely identify a single record within that table.

Next, groups of repeated fields should be eliminated. A good indication of a repeated field is a series of fields with the same root name followed by a different number. To remove the repeating group of fields, collapse them into a single field with multiple records in a new table, related back to the primary data.

Example:

This is our set of related attributes for site observations. First, identify the attributes that make up the unique key. Combining the site number and observation date will create a unique identifier for this table; together, they make up the primary key.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	Site Observation Data 1999-2002														
2	site_num (PK1)	park_code	latitude_deg	longitude_deg	elevation_ft	observer_name	observer_dept	observer_phone_num	observation_date (PK2)	tree1_species_code	tree1_height_m	tree2_species_code	tree2_height_m	tree3_species_code	tree3_height_m
3	2301102	YUCH	69.32677	-125.69411	700	LEM	BIO	9075552032	1999-07-05	BETPAR	2	PICGLA	4	PICMAR	5
4	1850201	YUCH	62.2554	-135.56671	2000	DEL	BIO	9075552033	2000-06-15	PICMAR	6	BETPAR	19	PICGLA	2
5	1850201	YUCH	62.2554	-135.56671	2000	ALS	GIS	9075550001	2002-01-20	PICMAR	6	BETPAR	20		
6	2301102	YUCH	69.32677	-125.69411	700	ALS	GIS	9075550001	2002-01-21	BETPAR	3	PICGLA	6	PICMAR	6

Recommended Database Strategies

Example: (continued)

Now, find the groups of repeating attributes. In this table, the tree species code and tree height are repeated multiple times per record. The clue for detecting these repeated fields is seeing tree1, tree2, and tree3 at the beginning of the field names. This indicates that multiple trees can be observed during a single site observation; these fields need to be separated into their own table that can be related back to the site observation data. The following steps can be followed to complete this separation.

1. Decide on the subject and name for the new table. This table will contain tree data, so it is named **tbl_tree**.
2. Decide on the attributes for the new table. The first repeating group is tree1_species_code, tree2_species_code, and tree2_species_code. The other repeating group contains the three tree height fields. These are the only fields in the original table related to trees, so the new table will contain the tree species and tree height, named tree_species_code and tree_height_m.
3. Identify the keys for the new table. Since this is a **child table** of site observation, the unique identifier from the **parent table** needs to be carried over into the tree table so the two tables remained linked together; site_num and observation_date are added as **foreign keys**. Since more than one tree of the same species can be observed in a single site observation, a sequential, auto-number is added for the primary key, named tree_id.

Here is the modified site observation table, along with the new tree table. These tables contain attributes related to the table subject and are identified with primary keys. There are no repeated fields. This data model is in first normal form.

	A	B	C	D	E	F	G	H	I
1	Site Observation Data 1999-2002								
2	site_num (PK1)	park_code	latitude_deg	longitude_deg	elevation_ft	observer_name	observer_dept	observer_phone_num	observation_date (PK2)
3	2301102	YUCH	69.32677	-125.69411	700	LEM	BIO	9075552032	1999-07-05
4	1850201	YUCH	62.2554	-135.56671	2000	DEL	BIO	9075552033	2000-06-15
5	1850201	YUCH	62.2554	-135.56671	2000	ALS	GIS	9075550001	2002-01-20
6	2301102	YUCH	69.32677	-125.69411	700	ALS	GIS	9075550001	2002-01-21

	A	B	C	D	E
1	tbl_tree				
2	tree_id (PK)	site_num (FK1)	observation_date (FK2)	tree_species_code	tree_height_m
3	1	2301102	1999-07-05	BETPAR	2
4	2	2301102	1999-07-05	PICGLA	4
5	3	2301102	1999-07-05	PICMAR	5
6	4	1850201	2000-06-15	PICMAR	6
7	5	1850201	2000-06-15	BETPAR	19
8	6	1850201	2000-06-15	PICGLA	2
9	7	1850201	2002-01-20	PICMAR	6
10	8	1850201	2002-01-20	BETPAR	20
11	9	2301102	2002-01-21	BETPAR	3
12	10	2301102	2002-01-21	PICGLA	6
13	11	2301102	2002-01-21	PICMAR	6

Note that the two fields that make up the primary key in the site observation table correspond exactly with the two foreign keys in the tree table. The tree table needs these two additional fields so that records in the tree table can be linked back to a record in the site observation table.

Signs of a Database in First Normal Form

- ✓ Each table is identified with a primary key
- ✓ No table contains repeating fields

Recommended Database Strategies

Step 2: Second Normal Form – Eliminate redundant data

The next step in normalizing your data is to get it into **second normal form**. Each attribute must depend on the entire primary key, so groups of repeated values are moved into separate, related tables. A good indication of redundant data is an attribute that depends on only part of a multi-field primary key.

Again, the child table, usually the “many” part of a one-to-many relationship must contain the primary key from the parent table. This is called a **foreign key** and allows the two tables to be linked together.

Example:

In the site observation table, there are two attributes making up the unique key. Find the attributes that are not dependent on both the site number and the observation date. In this table, there are some fields that are related only to the site and are not dependent on the observation (when and who); these fields need separated into their own table that can be related back to the site observation table. The following steps can be followed to complete this separation.

1. Decide on the subject and name for the new table. This table will contain site data, so it is named `tbl_site`.
2. Decide on the attributes for the new table. This table will contain attributes that are dependent only on the site number and not the observation date, so the new table will contain the park code, latitude, longitude, and elevation. The attribute names will not change.
3. Identify the keys for the new table. Since this is a parent table of site observation, the unique identifier will be a subset of the child table; `site_num` is added as the primary key. Since this table needs to link to its child table through the site number, the primary key of the observation table remains the same, although `site_num` is now a primary key and a foreign key (which indicates a relationship).

All attributes in the tree table are dependent on the primary key, so no changes are necessary.

Here is the modified observation table, the tree table, and the new site table. All attributes in these tables are dependent on the primary key of the table. This data model is in second normal form.

	A	B	C	D	E
1	tbl_site				
	site_num (PK)	park_code	latitude_deg	longitude_deg	elevation_ft
2					
3	2301102	YUCH	69.32677	-125.69411	700
4	1850201	YUCH	62.2554	-135.56671	2000

	A	B	C	D	E
1	tbl_observation				
	site_num (PK1, FK)	observer_name	observer_dept	observer_phone_num	observation_date (PK2)
2					
3	2301102	LEM	BIO	9075552032	1999-07-05
4	1850201	DEL	BIO	9075552033	2000-06-15
5	1850201	ALS	GIS	9075550001	2002-01-20
6	2301102	ALS	GIS	9075550001	2002-01-21

	A	B	C	D	E
1	tbl_tree				
	tree_id (PK)	site_num (FK1)	observation_date (FK2)	tree_species_code	tree_height_m
2					
3	1	2301102	1999-07-05	BETPAR	2
4	2	2301102	1999-07-05	PICGLA	4
5	3	2301102	1999-07-05	PICMAR	5
6	4	1850201	2000-06-15	PICMAR	6
7	5	1850201	2000-06-15	BETPAR	19
8	6	1850201	2000-06-15	PICGLA	2
9	7	1850201	2002-01-20	PICMAR	6
10	8	1850201	2002-01-20	BETPAR	20
11	9	2301102	2002-01-21	BETPAR	3
12	10	2301102	2002-01-21	PICGLA	6
13	11	2301102	2002-01-21	PICMAR	6

Recommended Database Strategies

Signs of a Database in Second Normal Form

- ✓ No table contains redundant data; or groups of repeated values for multiple records

Step 3: Third Normal Form – Eliminate columns not dependent on the key

The final step that you will complete to normalize your data is to get it into **third normal form**. Each attribute must depend on the primary key, so the violating fields are moved into separate, related tables. A good indication of non-dependent data is an attribute that is not directly related to the subject of the table.

Example:

In the observation table, there are two attributes making up the unique key. Find the attributes that are not related to the site number and observation date. In this table, there are some fields that are related to the observer, but not when the observation took place; these fields need separated into their own table that can be related back to the observation table. The following steps can be followed to complete this separation.

1. Decide on the subject and name for the new table. This table is a reference table for observers, so it is named ref_observer.
2. Decide on the attributes for the new table. This table will contain attributes that are dependent only on the observer name and not the observation date, so the new table will contain the observer name, department, and phone number. The attribute names will not change.
3. Identify the keys for the new table. The primary key for this table is observer_name. Since this is a reference table to the observation, the observer name in the observation table is now a foreign key (which indicates a relationship).

All attributes in the site and tree tables are dependent on the primary key, so no changes are necessary.

Here is the modified observation table, the tree table, the site table, and the new observer table. All attributes in these tables are dependent on the primary key of the table. This data model is in third normal form.

	A	B	C	D	E
1	tbl_site				
	site_num (PK1)	park_code	latitude_deg	longitude_deg	elevation_ft
2	2301102	YUCH	69.32677	-125.69411	700
3	1850201	YUCH	62.2554	-135.56671	2000

	A	B	C
1	tbl_observation		
	site_num (PK1, FK1)	observation_date (PK2)	observer_name (FK2/REF)
2	2301102	1999-07-05	LEM
3	1850201	2000-06-15	DEL
4	1850201	2002-01-20	ALS
5	2301102	2002-01-21	ALS

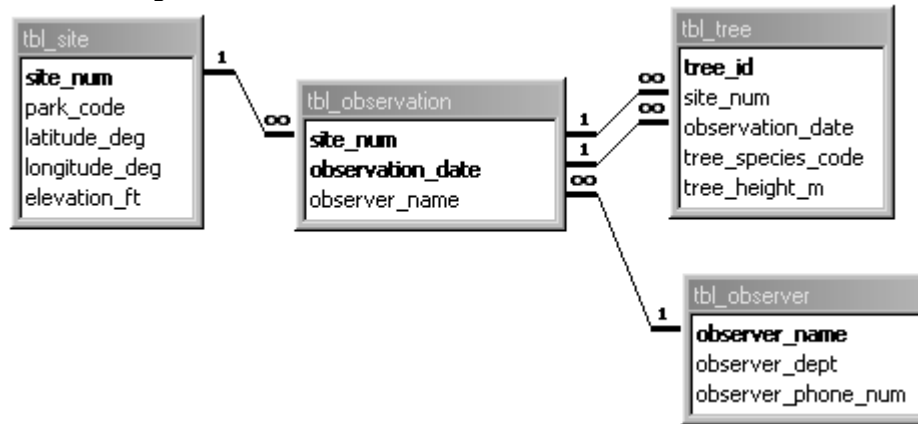
	A	B	C	D	E
1	tbl_tree				
	tree_id (PK)	site_num (FK1)	observation_date (FK2)	tree_species_code	tree_height_m
2	1	2301102	1999-07-05	BETPAR	2
3	2	2301102	1999-07-05	PICGLA	4
4	3	2301102	1999-07-05	PICMAR	5
5	4	1850201	2000-06-15	PICMAR	6
6	5	1850201	2000-06-15	BETPAR	19
7	6	1850201	2000-06-15	PICGLA	2
8	7	1850201	2002-01-20	PICMAR	6
9	8	1850201	2002-01-20	BETPAR	20
10	9	2301102	2002-01-21	BETPAR	3
11	10	2301102	2002-01-21	PICGLA	6
12	11	2301102	2002-01-21	PICMAR	6

	A	B	C
1	ref_observer		
	observer_name (PK)	observer_dept	observer_phone_num
2	LEM	BIO	9075552032
3	DEL	BIO	9075552033
4	ALS	GIS	9075550001

Recommended Database Strategies

Example: (continued)

Here is a diagram of the final database model.



Signs of a Database in Third Normal Form

- ✓ Each table contains only fields that are dependent on the primary key; or directly related to the subject of the table

Other Normal Forms

The remaining three normal forms are not required for your database. Here is an overview of them, for informational purposes. After Third Normal Form, comes Boyce-Codd Normal Form. For a database to pass this stage, every field must be a candidate key, or a contender for primary key. In Fourth Normal Form, no table can contain more than one multi-valued field. Fifth Normal Form basically covers everything else. In each of the previous normal forms, we are dividing the tables and removing fields that may cause data ambiguities or anomalies into separate tables. In this stage, you continue to divide tables in a similar manner until further splitting would result in tables that can not be joined to recreate the original or until the only splits that remain are trivial. As mentioned above, following normalization to such a fine degree can often make the database more difficult to work with. Hence, we do not require your database to go beyond Third Normal Form.

Recommended Database Strategies

Spatial Data – ArcView Shape Files

Attributes of a shape file are stored in a DBF. ArcView is not inherently a relational database management system. And because of this, it has limitations when working with tabular data, namely the same issues that occur when working with any type of flat file.

You are **strongly** encouraged to limit the fields you use as attributes when you generate shape files. Duplicating the entire dataset is not good practice because this leads to poor data integrity. If you have two copies of the dataset and either one can be updated, then you are allowing an opportunity for the copies to become out of sync. Chances are that when someone updates one copy in either Access or ArcView, the user will not follow through and update the other dataset. A much better alternative is to only extract required fields for your shape file. At a minimum, this can be the primary key, or unique identifier for that table. You may also want to include geographical oriented fields, such as latitude and longitude. With the release of the ArcView to Access product, the process of linking between your two datasets becomes simple and automated.

Note that you can use the linking feature in ArcView to view related data (in an ODBC compliant database) that is not stored with the shape file attributes. This gives the appearance of having all fields from your Access table as attributes. This method is an acceptable way of viewing the related fields because the fields are not actually copied into ArcView. Rather, there is a link that dynamically displays the data from Access alongside the shape file attributes. Refer to the [SQL Connect](#) and [Joining a database table to an ArcView table](#) topics in your ArcView help.

Recommended Database Strategies

The Database Template

The database template is a reusable set of tables and fields for the inventory and monitoring program. There are two kinds of tables.

tbl_study			
network_code	<pi>	A4	<M>
project_type	<pi>	VA10	<M>
protocol_code		VA10	
project_name		VA100	
project_desc		VA255	
park_lead_first_name		VA20	
park_lead_last_name		VA20	
park_lead_phone_num		A10	
princ_invest_first_name		VA20	
princ_invest_last_name		VA20	
princ_invest_phone_num		A10	
data_contact_first_name		VA20	
data_contact_last_name		VA20	
data_contact_phone_num		A10	
site_id_formula_desc		VA255	
site_location_desc		VA255	
location1_desc		VA100	
location2_desc		VA100	
location3_desc		VA100	
location4_desc		VA100	
observation_id_formula_desc		VA255	
study_pk	<pi>		

tbl_observation			
observation_id	<pi>	VA20	<M>
observer_name		VA20	
obs_date		D	
obs_start_time		T	
obs_end_time		T	
cloud_cover_percent		SI	
cloud_cover_code		VA20	
wind_speed_mph		SF	
wind_speed_code		VA20	
precipitation_code		VA20	
temperature_deg		SF	
temperature_code		VA20	
water_code		VA20	
noise_code		VA20	
disturbance_code		VA20	
observation_desc		VA255	
obs_pk	<pi>		

tbl_site			
site_id	<pi>	VA20	<M>
park_code		A4	
latitude_dec_deg		LF	
longitude_dec_deg		LF	
latitude2_dec_deg		LF	
longitude2_dec_deg		LF	
meridian_code		A1	
township_num		SI	
township_direction_code		A1	
range_num		SI	
range_direction_code		A1	
section_num		SI	
qtr_section_num		SI	
qtr_qtr_section_num		SI	
point_num		SI	
utm_zone_num		SI	
utm_x_coord		LF	
utm_y_coord		LF	
state_plane_zone_num		SI	
state_plane_x_coord		LF	
state_plane_y_coord		LF	
block_num		SI	
line_num		SI	
grid_name		VA10	
plot_num		SI	
transect_num		SI	
usgs_quad_name		VA50	
usgs_quad_num		VA10	
location1_value		VA20	
location2_value		VA20	
location3_value		VA20	
location4_value		VA20	
location_desc		VA255	
aspect_deg		SF	
slope_deg		SF	
elevation_height_m		SF	
landcover_class		VA20	
deu_code		VA20	
datum_code		VA20	
gps_type		VA20	
est_horz_error_dist_m		SF	
est_vert_error_height_m		SF	
accuracy_desc		VA255	
site_desc		VA255	
site_pk	<pi>		

The required **core** tables are common to all studies. They answer the basic question: *When and where was the data collected?* Each piece of data has a location of collection and each location has a set of characteristics that uniquely describe that point geographically. Similarly, each piece of data has a date and time of collection and a set of characteristics that describe the physical conditions of the location at that point in time. The layouts of these tables are static and allow all collected data to be summarized at the program level. With few exceptions, these tables will be imported into your database as-is and should not be modified. An additional study table contains one record of information about the dataset.

The **optional** tables are specific to individual studies. Rather than being exact tables that will be imported into your database, as with the required tables, these tables act as libraries of sample fields for each type of study. Users can pick and choose from these tables anything that applies to their study. These tables are dynamic; they will develop more robustly over time, changing as users append additional useful information. These tables will only be limited by the contributions of the users. Examples of optional table data include habitat, small mammals, fish, etc.

To implement the database template, you will need a unique identifier for each data collection site. If your database already contains such a reference, that value may be loaded directly into the site_id field. You must also indicate the location using one of the geographical references in the table.

You will need a unique identifier for each observation. If your database already contains such a reference, that value may be loaded directly into the observation_id field. In other cases, we suggest you use the date. Generally, the observation date and the site identifier will be a unique combination. The date should be formatted, since this is a text field (even though this goes against our rule regarding formatting). Use the format YYYYMMDD where YYYY is the four-digit year, MM is the two-digit month, and DD is the two-digit day.

All fields (minus some of the geographical location fields as explained below) in these required tables are highly recommended. However, we do realize that some field studies are complete and the database fields cannot be populated if the data was not collected during the site observations.

Required Tables of the Database Template

Recommended Database Strategies

Study Table

This table supplies important information about the overall study project. This table is required to synchronize your database with the national database template. When completed, this table will contain only a single record.

Field Name	Data Type	Length	Is Required	Validation Rules	Description
network_code	text	4	yes	= [AK, CAN, NWAN, SEAN, SWAN]	This field contains the Alaska network in which the study is conducted. This value will be used to generate a unique site and observation when rolled up at the national level.
project_type	text	10	yes	= [BIRDS, FISH_FRESH, FISH_MRINE, PLANTS, SM_MAMMALS]	This field contains the study focus. This value will be used to generate a unique site and observation when rolled up at the national level.
protocol_code	text	10	no		This field contains the name and/or version of the protocol used. The protocol should be directly associated with the project code.
project_name	text	100	no		This field contains the full project name.
project_desc	text	255	no		This field contains a brief description of the project.
park_lead_first_name	text	20	no		This field contains the first name of the park lead for the project.
park_lead_last_name	text	20	no		This field contains the last name of the park lead for the project.
park_lead_phone_num	text	10	no		This field contains the phone number of the park lead for the project.
princ_invest_first_name	text	20	no		This field contains the first name of the principal investigator for the project.
princ_invest_last_name	text	20	no		This field contains the last name of the principal investigator for the project.

Recommended Database Strategies

Field Name	Data Type	Length	Is Required	Validation Rules	Description
princ_invest_phone_num	text	10	no		This field contains the phone number of the principal investigator for the project.
data_contact_first_name	text	20	no		This field contains the first name of the data contact for the project.
data_contact_last_name	text	20	no		This field contains the last name of the data contact for the project.
data_contact_phone_num	text	10	no		This field contains the phone number of the data contact for the project.
site_id_forumla_desc	text	255	yes		This field contains a detailed description of the formula used to generate site ids for the project.
site_location_desc	text	255	no		This field contains a description of the site fields used to identify the geographic location.
location1_desc**	text	100	no		If the location1_value field is used in the site table, this field contains a description of the stored values and their source.
location2_desc**	text	100	no		If the location2_value field is used in the site table, this field contains a description of the stored values and their source.
location3_desc**	text	100	no		If the location3_value field is used in the site table, this field contains a description of the stored values and their source.
location4_desc**	text	100	no		If the location4_value field is used in the site table, this field contains a description of the stored values and their source.
observation_id_formula_desc	text	255	yes		This field contains a detailed description of the formula used to generate observation ids for the project.

Recommended Database Strategies

****** Note that these fields, along with a few others throughout the database template do conflict with our rules of normalization. Since the intention of the database template is accommodate data collected from all studies and we cannot envisage all of the geographical fields that a user may choose to describe a location, we have included these generic fields to cover those unpredictable cases.

Recommended Database Strategies

Site Table

Field Name	Data Type	Length	Is Required	Validation Rules	Description
site_id	text	20	yes		This field uniquely identifies a data collection location. The data in this field should be fixed width, that is all values should have the same format and length. The format for this field should be documented in the study header table. Ideally, this is the concatenation of the individual geographic location fields. Obviously this is not possible for all selections. If you are using lat/long, utm, or state plane coordinates, please derive a numbering system for your collection sites, preferably a naming convention meaningful to the study. A 4-char numeric sequence (0000-9999) is least desirable, but should be used if you are not able to create a better numbering system.
park_code	text	4	yes	ref_park	This field contains the 4-character NPS unit code for the park in which the data collection location exists. A list of valid values will be maintained in the reference table, ref_park.
latitude_dec_deg longitude_dec_deg latitude2_dec_deg longitude2_dec_deg	double double double double	- - - -	yes *at least one of these sets of geographic location fields are required		These fields contain the latitude and longitude of the data collection location, in decimal degrees. If the data collection location represents a line, this point should identify the starting point. If the data collection location represents a polygon, this point should identify the centroid of the plot. If the latitude and longitude are chosen as the geographic location fields, there is no need to populate utm or state plane coordinates. These values may be derived at a later data, if necessary. You will need to derive a site id.

Recommended Database Strategies

Field Name	Data Type	Length	Is Required	Validation Rules	Description
meridian_code	text	1	yes*	= [C, F, K, S, U]	These fields contain the meridian, township, range, and section of the data collection location. The point number can further identify points within the mtrs. If this data is collected, we recommend using it as the site id in the following format: MTTTTRRRRSSPPP where M is the 1-char meridian, TTTT is the township number padded out to 3-char plus the 1-char township direction, RRRR is the range number padded out to 3-char plus the 1-char range direction, SS is the section padded out to 2-char, and PPP if used, is your sequential point number padded out to 3-char. The corresponding site id may look something like this: F019N024E01 or S009N041W01001. If the quarter section or quarter-quarter section is used, they should be padded to the appropriate length and included in the site id. We recommend that you also collect the lat/long, utm, or state plane coordinates of the location.
township_num	integer	-		= [1...180]	
township_direction_code	text	1		= [N, S]	
range_num	integer	-		= [1...360]	
range_direction_code	text	1		= [E, W]	
section_num	integer	-		= [1...36]	
qtr_section_num	integer	-			
qtr_qtr_section_num	integer	-			
point_num	integer	-			
utm_zone_num	integer	-	yes*	= [1...52] or [2..10]	These fields contain the utm zone along with the x and y coordinates from the utm grid system. If the data collection location represents a line, this point should identify the starting point. If the data collection location represents a polygon, this point should identify the centroid of the plot. If the utm grid system is chosen to populate the geographic location fields, there is no need to populate lat/long or state plane coordinates. These values may be derived at a later data, if necessary. You will need to derive a site id.
utm_x_coord	double	-			
utm_y_coord	double	-			

Recommended Database Strategies

Field Name	Data Type	Length	Is Required	Validation Rules	Description
state_plane_zone_num state_plane_x_coord state_plane_y_coord	integer double double	- - -	yes*	= [1...52] or [2..11]	These fields contain the state plane zone along with the x and y coordinates from the state plane coordinate system. If the data collection location represents a line, this point should identify the starting point. If the data collection location represents a polygon, this point should identify the centroid of the plot. If the state plane coordinate system is chosen to populate the geographic location fields, there is no need to populate lat/long or utm coordinates. These values may be derived at a later data, if necessary. You will need to derive a site id.
block_num line_num point_num	integer integer integer	- - -	yes*	= [1...999] = [1...99] = [1...99]	These fields contain the block, line, and point of the data collection location. If this data is collected, we recommend using it as the site id in the following format: BBBLLPP where BBB is the block number padded out to 3-char, LL is the line number padded out to 2-char, PP is the point number padded out to 2-char. The corresponding site id may look something like this: 1620101 or 1651002. We recommend that you also collect the lat/long, utm, or state plane coordinates of the location.
grid_name point_num	text integer	10 -	yes*		These fields contain the grid name and the point number, which further defines a location within the grid. If this data is collected, we recommend using it as the site id in the following format: GGGGGGPPP where GGGGGG is the grid name abbreviated to 6-char and PPP is the point number padded out to 3-char. The corresponding site id may look something like this: ROCKRV003 or STRLCR088.

Recommended Database Strategies

Field Name	Data Type	Length	Is Required	Validation Rules	Description
plot_num transect_num point_num	integer integer integer	- - -	yes*		These fields contain the plot number, transect number, and point number of the data collection location. If this data is collected, we recommend using it as the site id in the following format: LLLLLTTPP where LLLLL is the plot number, TT is the transect number padded out to 2-char, and PP is the point number padded out to 2-char. The corresponding site id may look something like this: 451880101 or 499330512.
usgs_quad_name usgs_quad_num	text text	50 10	yes*		These fields contain the USGS quad name and number. You will need to derive a site id.
location1_value location2_value location3_value location4_value location_desc	text text text text text	20 20 20 20 255	yes*		These fields are left in a generic form for a location identifying convention that falls outside those previously defined in this table. You may use up to three fields to identify your location. These fields are defined as text. If you are storing numeric values, feel free to change the data type appropriately. Please document your location identification strategy in the study header table.
aspect_deg	single	-	no	[0...360]	This field contains the aspect of the data collection point, in degrees clockwise from true north.
slope_deg	single	-	no		This field contains the slope of the data collection point, in degrees.
elevation_height_m	single	-	no		This field contains the elevation of the data collection location, in meters.
landcover_class	text	20	no	ref_landcover	This field contains the elevation of the landcover class. A list of valid values will be maintained in the reference table, ref_landcover.

Recommended Database Strategies

Field Name	Data Type	Length	Is Required	Validation Rules	Description
deu_code	text	20	no	ref_deu	This field contains the detailed ecological unit. A list of valid values will be maintained in the reference table, ref_deu.
datum_code	text	20	no	ref_datum	This field contains the reference system used for defining the coordinates of points. A list of valid values will be maintained in the reference table, ref_datum.
gps_type	text	20	no	ref_gps_type	This field contains the type of gps unit used to collect data. A list of valid values will be maintained in the reference table, ref_gps_type.
est_horz_error_dist_m	double	-	no		This field contains the "error buffer" associated with the x, y coordinates of the data collection location, in meters.
est_vert_error_height_m	double	-	no		This field contains the vertical error associated with the elevation of the location, in meters.
accuracy_desc	text	255	no		This field contains any notes related to the positional accuracy of the coordinates.
site_desc	text	255	no		This fields contains a brief description of the data collection point, uniquely identified by the site id.

Recommended Database Strategies

Observation Table

Field Name	Data Type	Length	Is Required	Validation Rules	Description
observation_id	text	20	yes		This field, with the site id, uniquely identifies a data collection observation. The data in this field should be fixed width, that is all values should have the same format and length. The format for this field should be documented in the study header table. If you have a convention for uniquely identifying an observation, use that here. Otherwise, we recommend using the observation date. If only one observation can occur at a single location in a single day, format the date as YYYYMMDD where YYYY is the 4-char year, MM is the month padded to 2-char, and DD is the day padded to 2-char. You may also choose to format the date as YYYYJJJ where JJJ is the julian date padded to 3-char. If multiple observations can occur at the same location in a single day, you should append a sequence padded to 2 or 3-char (001, 002, 003...) or the start time in the format HHMM where HH is the hour (0-23) padded to 2-char and MM is the minute padded to 2-char.
site_id	text	20	yes		This field, with the observation id, uniquely identifies a data collection observation. This site id must already exist as a valid data collection location in the site table.
observer_name	text	20	yes		This field contains the name or initials of the primary observer. If it is necessary to record multiple observers, you may create a linked table where a single observation can have many observers.
obs_date	date/time	-	yes		This field contains the observation date.

Recommended Database Strategies

Field Name	Data Type	Length	Is Required	Validation Rules	Description
obs_start_time	date/time	-	no		This field contains the start time of the observation or sampling.
obs_end_time	date/time	-	no		This field contains the end time of the observation or sampling.
cloud_cover_percent	integer	-	no		This field contains actual percentage of cloud cover at the time of observation. This field is preferred over the cloud cover range.
cloud_cover_code	text	20	no	ref_cloud_cover	This field contains a range for the cloud coverage at the time of observation. The cloud cover percent field is preferred over this.
wind_speed_mph	single	-	no		This field contains the actual wind speed at the time of observation. This field is preferred over the wind speed range.
wind_speed_code	text	20	no	ref_wind_speed	This field contains a range for the wind speed at the time of observation. The actual wind speed is preferred over this.
precipitation_code	text	20	no	ref_precipitation	This field contains the precipitation present at the time of observation. A list of valid values will be maintained in the reference table, ref_precipitation.
temperature_degf	single	-	no		This field contains the actual temperature at the time of observation. This field is preferred over the temperature range.
temperature_code	text	20	no	ref_temperature	This field contains a range for the temperature at the time of observation. The actual temperature is preferred over this.
water_code	text	20	no	ref_water	This field contains the bodies of water present at the site and time of observation. A list of valid values will be maintained in the reference table, ref_water.

Recommended Database Strategies

Field Name	Data Type	Length	Is Required	Validation Rules	Description
noise_code	text	20	no	ref_noise	This field contains the level of noise present at the time of observation. A list of valid values will be maintained in the reference table, ref_noise.
disturbance_code	text	20	no	ref_disturbance	This field contains any disturbance present at the time of observation. A list of valid values will be maintained in the reference table, ref_disturbance.
observation_desc	text	255	no		This fields contains a brief description of the observation, uniquely identified by the observation id and site id.

Recommended Database Strategies

How to Implement the Database Template and Naming Standards

You may be in the process of defining the data you will collect and have not yet completed your field study. Or you may have already created your relational database and data entry forms and are in the process of entering data that is already collected. Where your project sits on this scale determines to what degree you will implement the database template and naming standards recommended in this document.

The following chart will advise you of the steps most suitable for your project. Review the left side of the chart. Find all milestones that have been completed for your project and select the single row that most accurately matches the current state of your project. Follow that row across the chart; the steps that you need to complete for this project are checked. Complete those appropriate steps to implement the database template and naming standards for your project.

Milestones Completed for Your Project							Steps to Follow for Implementing the Database Template and Naming Standards						
Data collection fields are defined	Data collection is complete	Data is entered in a flat file format	Database tables are created	Data is entered in a relational database	Forms are created from the wizard	Custom forms are created	Step 1	Step 2	Step 3	Step 4	Step 5	Step 6	Step 7
							Copy the database template to your Access database	Design your database around the database template and to observe the naming standards	Complete your field data collection	Rename your tables and fields to observe the naming standards	Map your fields into the database template	Enter or convert your data into Access E = Enter C = Convert	Populate the core tables of the database template
■	■		■	■		■	✓				✓		✓
■	■		■	■	■		✓			✓	✓		✓
■	■		■	■			✓			✓	✓		✓
■	■		■				✓	✓			✓	✓ _E	
■	■	■	■				✓	✓			✓	✓ _C	
■	■	■					✓	✓			✓	✓ _C	
■	■						✓	✓			✓	✓ _E	
■							✓	✓	✓			✓ _E	
							✓	✓	✓			✓ _E	

Recommended Database Strategies

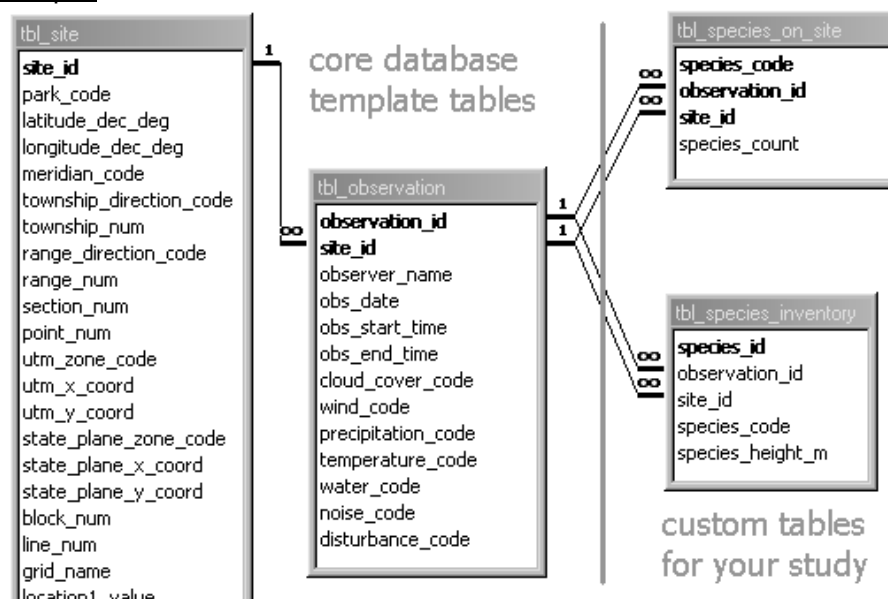
Step 1: Copy the database template

Regardless of your project status, you will need to import the most recent version of the core database template tables into your database.

Step 2: Design your relational database

You will complete this design step if you have not yet designed your database or even if you have created your database tables, but they are not yet populated. Begin by making sure you have the latest core tables from the database template (step 1). The site and observation tables are the foundation of your database. All other data tables should link to the observation table. The primary key of the observation table is the combination of the site_id and observation_id fields. These fields will be carried over as foreign keys in the new tables to link back to the observation table. They may or may not be part of the primary key in the new tables, depending on the table design.

Example:



The upper custom table records a summary of plant species found during an observation at a particular site. Since each species will be recorded only once per observation and site, the species_code is included as part of the primary key along with observation_id and site_id. In this case, the observation_id and site_id are part of the primary key for this species table, and foreign keys that link this table back to the correct observation in that table.

The lower custom table records each and every plant found during an observation at a particular site. Since each species could be recorded multiple times per observation and site, a sequential auto-number field has been added, named species_id. This field could be part of a multi-field primary key, as in the example above, but species_id is unique on its own, so only that field has been chosen as the primary key. In this case, observation_id and site_id are only foreign keys that link this table back to the correct observation.

If you have already created your database tables (but they are not populated), then the right side of this diagram is already complete in your database. You will need to modify your existing tables to link to the core database template tables. This step includes adding observation_id and site_id to your tables as appropriate and in some cases, changing the primary key of your tables. In addition to reworking your database to incorporate the template, you will also need to rename your fields to fit with the naming convention standards (step 4).

Approval of the database design, to ensure that it meets the study plan and the adopted naming convention standards, may be required by your data manager.

Recommended Database Strategies

Step 3: Complete your field data collection

Step 4: Rename your tables and fields

You will complete this step if you have already created your database tables, and have not created custom forms. If you have forms that were generated through the Access wizard and you have not spent much time customizing them, you should rename all your tables and fields and then regenerate the forms. When renaming your fields, follow the naming convention standards described in the **Table Name** and **Field Names** sections of this document.

Step 5: Map your fields to the database template

If you've already collected your field data, complete the following charts to create a mapping from your database to the matching fields in the database template. Refer back to previous section for field definitions. The fields in bold are mandatory; you must have a value for these fields. Generally, the mapping will contain the name of a table and field from your data. However, some values may be the same for all records in the study. In these cases, you may enter an actual value into the mapping chart.

Example:

Field Name	Data Type	Field(s) in Your Access Database
site_id	text(20)	tbl_study_site_observation.site_id
park_code	text(4)	DENA
latitude_dec_deg	double	tbl_study_site_observation.latitude
longitude_dec_deg	double	tbl_study_site_observation.longitude

Field Name	Data Type	Field(s) in Your Access Database
site_id	text(20)	tbl_study_site_observation.site_id
park_code	text(4)	tbl_study_site_observation.park
latitude_dec_deg	double	tbl_study_site_observation.latitude
longitude_dec_deg	double	tbl_study_site_observation.longitude

If your study was restricted to a single park, then all site records in your study will have the same park_code. You may not have collected this as part of your field study, since the value was known and the same for all observation sites. In this case, you can enter the actual park code, such as DENA, in the space across from park_code.

If your study took place among multiple parks, then you should have collected that data along with other site information. In this case, you would enter the name of the name and field from your database that stores the park abbreviation.

Site Table

Field Name	Data Type	Field(s) in Your Access Database
site_id	text(20)	
park_code	text(4)	
latitude_dec_deg	double	
longitude_dec_deg	double	

Recommended Database Strategies

Field Name	Data Type	Field(s) in Your Access Database
latitude2_dec_deg	double	
longitude2_dec_deg	double	
meridian_code	text(1)	
township_num	integer	
township_direction_code	text(1)	
range_num	integer	
range_direction_code	text(1)	
section_num	integer	
qtr_section_num	integer	
qtr_qtr_section_num	integer	
point_num	integer	
utm_zone_num	integer	
utm_x_coord	double	
utm_y_coord	double	
state_plane_zone_num	integer	
state_plane_x_coord	double	
state_plane_y_coord	double	
block_num	integer	
line_num	integer	
grid_name	text(10)	
plot_num	integer	
transect_num	integer	
usgs_quad_name	text(50)	

Recommended Database Strategies

Field Name	Data Type	Field(s) in Your Access Database
usgs_quad_num	text(10)	
location1_value	text(20)	
location2_value	text(20)	
location3_value	text(20)	
location4_value	text(20)	
location_desc	text(255)	
aspect_deg	single	
slope_deg	single	
elevation_height_m	single	
landcover_class	text(20)	
deu_code	text(20)	
datum_code	text(20)	
gps_type	text(20)	
est_horz_error_dist_m	double	
est_vert_error_height_m	double	
accuracy_desc	text(255)	
site_desc	text(255)	

Observation Table

Field Name	Data Type	Field(s) in Your Access Database
observation_id	text(20)	
site_id	text(20)	
observer_name	text(20)	

Recommended Database Strategies

Field Name	Data Type	Field(s) in Your Access Database
obs_date	date/time	
obs_start_time	date/time	
obs_end_time	date/time	
cloud_cover_percent	integer	
cloud_cover_code	text(20)	
wind_speed_mph	single	
wind_speed_code	text(20)	
precipitation_code	text(20)	
temperature_degf	single	
temperature_code	text(20)	
water_code	text(20)	
noise_code	text(20)	
disturbance_code	text(20)	
observation_desc	text(255)	

Step 6: Enter or convert your data to Access

If your data is not yet in electronic format, then you will need to manually enter the data. This is best accomplished by creating data entry forms. Access is equipped with a form wizard that walks you through the steps of selecting form characteristics and then generates a form for you. You should complete this for each of the tables in your database. The wizard does not always create the most attractive form, but you can easily modify them to suit your tastes. Data entry is easiest if your forms imitate the relationships.

Example:

Recommended Database Strategies

The image shows three screenshots of database forms. The top-left form is 'tbl_site' with fields: site_id, utm_zon, location, est_vest, park_cd, utm_x, location, accuracy, latitude, utm_y, aspect, site_desc, longitude, state_pl, slope_id, township, state_pl, elevation, township, block_r, landon, range_1, line_rn, den_co, range_2, grid_nu, datum, section, location, gpi_typ, point_n, location, est_hor. The bottom-left form is 'tbl_observation' with fields: observation_id, noise_code, site_id, disturbance_code, observer_name, cloud_cover_code, obs_date, wind_code, obs_start_time, precipitation_code, obs_end_time, temperature_code, water_code. The right form is a subform for 'tbl_observation' with fields: observation_id, noise_code, site_id, disturbance_code, observer_name, cloud_cover_code, obs_date, wind_code, obs_start_time, precipitation_code, obs_end_time, temperature_code, water_code.

For example, in the database template the site table is the parent to observation table because a single site can have multiple observations over time. You would begin by creating two forms; one for sites and one for observations. Then on the site form (because this is the parent), you can create a **subform**. This allows you to embed one form within another. Again a wizard will walk you through selecting characteristics. The site id is the linking field that allows the observation subform to work properly. Now you can enter all this information at once, rather than opening two different forms to enter related data.

If your data is already in an electronic format, such as a flat file, you may be able to convert your data into Access without reentering it all manually. Accomplish this by importing the flat file into Access then appending it, with translations into your database template core and study-specific tables.

Example:

The screenshot shows an Access 'Append Query' window. The query name is 'tbl_study_site_observation'. The fields listed are: site_id, park, latitude, longitude. Below the fields, there is a table with columns: Field, Table, Sort, Append To, Criteria, or. The table contains the following data:

Field	Table	Sort	Append To	Criteria	or
site_id	tbl_study_site_observation		tbl_study_site_observation	site_id	
park	tbl_study_site_observation		tbl_study_site_observation	park_code	
latitude	tbl_study_site_observation		tbl_study_site_observation	latitude_dec_deg	
longitude	tbl_study_site_observation		tbl_study_site_observation	longitude_dec_deg	

In this example, the flat file was imported into Access and named tbl_study_site_observation. Using an append query, we can map the fields from the flat file into the database tables. Since the flat file is not normalized, you must eliminate duplicates result records from the query prior to appending the records to your new tables. This is done by specifying a "distinct" query, which eliminates those duplicate records. Otherwise, an error will occur.

If all records in the flat file are for DENA, and there's no corresponding park field in your flat file, substitute **"DENA"** for **park** in the top row. This will insert the text value into the park_code field for all records in the table.

Step 7: Populate the database template

Regardless of your project status, you will need to populate the database template core tables in your database. If you incorporated the template into your database early on and the template is integrated with your data model, then data should have already been entered during the previous step. If these tables are not yet populated, you may be able automate this step with an append query, as described in the previous step.

Recommended Database Strategies

Adapting to the National Database Template

More information on the database template can be found on the intranet at <http://www.nature.nps.gov/im>. Please notice in the documentation that there are slight differences between the national template and the modified version we are adopting within the Alaska region. We have adapted the template to fit our standards for database naming conventions. We have also made changes to better fit our methods of data collection for locations. The following diagram shows the mapping from the Alaska template into the national.

tblLocations

Field in tblLocations	Data Type	Field(s) in Alaska Database Template
LocationID	char(255)	tbl_site.park_code + "_" + tbl_study.project_type + "_" + tbl_site.site_id
ParkCode	char(4)	tbl_site.park_code
Project	char(10)	tbl_study.project_type
Description	char(255)	tbl_site.site_desc
StartUTMX	double	tbl_site.utm_x_coord
StartUTMY	double	tbl_site.utm_y_coord
StopUTMX	double	n/a
StopUTMY	double	n/a
UTMZone	char	format(tbl_site.utm_zone_num, "00")
StartLat	double	tbl_site.latitude_dec_deg
StartLon	double	tbl_site.longitude_dec_deg
StopLat	double	n/a
StopLon	double	n/a
Datum	char	tbl_site.datum_code
DataType	char	n/a
EstHError	single	tbl_site.est_horz_err_dist_m
EstVError	single	tbl_site.est_vert_err_height_m

Recommended Database Strategies

Field in tblLocations	Data Type	Field(s) in Alaska Database Template
AccNotes	memo	<p>"Accuracy Desc = " + tbl_site.accuracy_desc</p> <p>"; Site Location Desc = " + tbl_study.site_location_desc</p> <p>If the MTRS has been collected:</p> <p>"; Meridian = " + tbl_site.meridian_code + "; Township = " + tbl_site.township_num + tbl_site.township_direction_code + "; Range = " + tbl_site.range_num + tbl_site.range_direction_code + "; Section = " + tbl_site.section_num + tbl_site.point_num</p> <p>If state plane coordinates have been collected:</p> <p>"; State Plane Zone = " + tbl_site.state_plane_zone_num + "; X Coord = " + tbl_site.state_plane_x_coord + "; Y Coord = " + tbl_site.state_plane_y_coord</p> <p>If the block, line, and point have been collected:</p> <p>"; Block = " + tbl_site.block_num + "; Line = " + tbl_site.line_num + "; Point = " + tbl_site.point_num</p> <p>If the grid and point have been collected:</p> <p>"; Grid = " + tbl_site.grid_name + "; Point = " + tbl_site.point_num</p> <p>If the plot, transect, and point have been collected:</p> <p>"; Plot = " + tbl_site.plot_num + "; Transect = " + tbl_site.transect_num + "; Point = " + tbl_site.point_num</p> <p>If a miscellaneous convention has been used:</p> <p>"; " + tbl_study.location1_desc + " = " + tbl_site.location1_value + "; " + tbl_study.location2_desc + " = " + tbl_site.location2_value + "; " + tbl_study.location3_desc + " = " + tbl_site.location3_value</p> <p>For all sites:</p> <p>"; Aspect = " + tbl_site.aspect_deg + "; Slope = " + tbl_site.slope_deg + "; Elevation = " + tbl_site.elevation_height_m</p> <p>"; Site ID Formula = " + tbl_study.site_id_formula_desc</p> <p>"; Observation ID Formula = " + tbl_study.observation_id_formula_desc</p>

Recommended Database Strategies

tblEvents

Field in tblEvents	Data Type	Field(s) in Alaska Database Template
EventID	char	tbl_site.park_code + "_" + tbl_study.project_type + "_" + tbl_site.site_id + "_" + tbl_observation.observation_id
Project	char(10)	tbl_study.project_type
Year	date	Format(tbl_observation.obs_date, "YYYY")
StartDate	date	tbl_observation.obs_date
EndDate	date	n/a
StartTime	char?	Format(tbl_observation.obs_start_time, "HH:MM")
EndTime	char?	Format(tbl_observation.obs_end_time, "HH:MM")
TripReport	oleobject	n/a
Protocol	char	"AK_" + tbl_study.project_type + "_" + tbl_study.protocol_version_num

Guidelines to Database Design

In addition to implementing the database template, it is important that we standardize the databases as much as possible.

Standards are an important part of database design, as they allow for the development of consistent databases. This is extremely important when data is shared among multiple databases and potentially multiple database servers or when data is accessed and viewed by many users. Often, data from many separate studies are combined. If each individual database is developed using the same set of standards, the process of merging and comparing the data is much less complex than if databases are designed without using common rules and naming conventions. If users are familiar with the rules and guidelines, they can open any database developed with those standards and understand the setup of tables and fields. Over time, the contributions of each user can lead to a proficient and useful set of enterprise level data. Additionally, data entry windows and full-capability applications become easier to develop with a well-designed database.

Although the practice of implementing standards may be more time consuming the first design effort, each subsequent database that you create will become easier as you become more familiar with the rules and guidelines. With each database, your style will improve as you realize previous shortcomings. You will also find that portions of previously created databases can (and should!) be reused. Consistency is the key to high-quality organizational data.

Advantages of a Good Database Design

- ✓ You can easily modify and maintain the structure of your tables and fields
- ✓ You can easily modify the data in your database
- ✓ You can easily retrieve and compare information from your database
- ✓ You can easily develop data entry windows and more robust applications for your database

Recommended Database Strategies

Table Names

A good database design begins with intelligent naming of your tables and fields. Names should be designed such that anyone, even someone completely unfamiliar to your study and/or data, can look at your database and decipher its contents. This means assigning table and field names that clearly define the data being stored.

We will create our table names by answering two questions:

What category does this table fall into: data table, reference table, or cross-reference table?

What noun or short phrase summarizes the contents of this table?

How To Name a Table

Step 1: Select the prefix associated with the table category

Table Name Prefixes

Table Category	Prefix	Explanation	Examples
data table	tbl_	TBL is the abbreviation for table. This indicates that the table is one of the main tables containing data collected in the field	tbl_site contains data about the location of the field data collection tbl_sample_event contains the date and time of each observation event along with physical conditions present at that time
reference table	ref_	REF is the abbreviation for reference. This indicates that the table is a validation table and contains reference, or lookup data. Once populated, the data in these tables rarely changes.	ref_park contains a lookup list of all parks in the Alaska region ref_precipitation contains a lookup list of types of precipitation that could be taking place during an observation; valid values include none, fog/smoke, drizzle, showers, rain, sleet, or snow
cross-reference table	xref_	XREF is the abbreviation for cross-reference. This represents the advanced linking table and indicates a many-to-many relationship between two or more tables.	Each observer can collect on different days and in different locations. Each sample may be collected by more than one person. This qualifies as a many-to-many relationship between tbl_observer and tbl_sample_event. xref_sample_observer contains the unique identifier from each of its parent tables, thus joining or linking the two tables together

Recommended Database Strategies

Step 2: Select a noun or short phrase that summarizes the contents of the table

Follow the same rules described in the following section for field names. Additionally, a good table name...

- ✓ has a unique descriptive name that is meaningful to the entire organization
- ✓ accurately, clearly, and unambiguously identifies the subject of the table
- ✓ has the minimum number of words necessary to convey the subject of the table
- ✓ does not contain words that describe physical characteristics such as "file", "record", "data", or "table"
- ✓ does not contain unknown acronyms or abbreviations
- ✓ does not contain proper names and other words that limit the data that can be entered into the table
- ✓ does not implicitly or explicitly identify more than one subject
- ✓ does not use the plural form of a name

Step 3: Write a description for the table

Write a clear table description that accurately defines the table.

Recommended Database Strategies

Field Names

A good database design begins with intelligent naming of your tables and fields. Names should be designed such that anyone, even someone completely unfamiliar to your study and/or data, can look at your database and decipher its contents. This means assigning table and field names that clearly define the data being stored.

A good field name...

- ✓ has a unique descriptive name that is meaningful to the entire organization
- ✓ accurately, clearly, and unambiguously identifies the characteristic represented by the field
- ✓ has the minimum number of words necessary to convey the meaning of the characteristic the field represents
- ✓ is not a reserved word
- ✓ does not contain unknown acronyms or abbreviations
- ✓ does not contain words that confuse the meaning of the field name
- ✓ does not implicitly or explicitly identify more than one characteristic
- ✓ uses the singular form of the name

We will create our field names by answering three questions:

What noun or short phrase summarizes the contents of this field?

What category describes the type of data in this field?

What unit of measure, if any, is required for this field?

How to Name a Field

Step 1: Select a noun or short phrase that summarizes the contents of the field

For both table and field names, you are asked to come up with a noun or short phrase that summarizes the meaning. This is the root name and the most important part of the full table or field name. This name should be descriptive, but not too long. You can use abbreviations, but the name should not be so cryptic that others cannot figure it out. Spaces, special characters, and upper case letters should be avoided for technical reasons. Words should be separated with underscores for easier readability. Since Access is very lenient and allows easy implementation of any of these poor design choices, it is wholly up to the user to make better decisions. This table describes the rules in detail.

Root Table and Field Names

Recommended Database Strategies

Rule	Explanation	Importance	Examples of poor design	Examples of better design
Avoid spaces	Many users include spaces in table and field names for readability. Instead, use an underscore between words (Separate words rule). Access is one of the only database servers to even allow the use of spaces in names so other databases and programs cannot easily recognize your data. Any programming or customization becomes much more difficult when spaces are used.	High	Sample Events Sample ID	sample_event sample_id
Avoid special characters	Use only letters (and numbers, if necessary) in table and field names. Additionally, a name should never begin with a number. Again, Access is one of the only databases to even allow this. Any programming or customization becomes much more difficult when spaces are used.	High	Site/Locations phone# %MossCover Rain? project\$	site_location phone_num moss_cover_percent is_raining project_cost
Separate words	Many users choose mixed case table and field names for readability. Rather than relying upon the way the name is typed, a better choice is to use lowercase letters and separate the words with underscores (_). The name then resembles written text and words within the name are easily identified. Keep in mind that some database servers are case sensitive. This means that anyone who accesses data in your database has the type the names exactly as you did during creation. This can cause confusion and difficulty in using the data.	Medium	BirdDetections CloudCover dwarfsaplingheight	bird_detection cloud_cover dwarf_sapling_height_ft
Consistent case	Many users choose mixed case table and field names for readability. Rather than relying upon the way the name is typed, a better choice is to use an underscore between words (Separate words rule). Although you can use mixed case with underscores, a better choice is to be consistent in your case type (lowercase is preferred). Some databases are case sensitive and using mixed case will cause problems for others trying	Medium	DIGITAL_PHOTO SampleStartTime WindSpeed	digital_photo sample_start_time wind_speed_mph

Recommended Database Strategies

Rule	Explanation	Importance	Examples of poor design	Examples of better design
	to access your data because they have to type it exactly as you have created them.			
Singularize names	Always choose the singular noun form for names.	Medium	life_stages	life_stage
Avoid abbreviations	Avoid abbreviations unless the field name exceeds the maximum recommended limit (Limit length rule). If they are used, make every attempt to use known abbreviations (i.e. h2o for water rather than wat or wtr). In older flat file database systems, table and field names had a limit of 8 or 10 characters. Today's relational databases, including Access, do not impose these restrictive limits so the use of terse names and extensive abbreviating are not necessary. Take advantage of additional characters to eliminate the ambiguity of your field names.	Medium	smptrnid wtr_cd	sample_transect_id water_code
Limit length	Limit the length of table and field names to around 20 characters. This is done more for practicality than necessity. However, many database servers have a maximum length of 30, so names should not exceed 30 characters. Shorter names can be typed more quickly and are easier to remember. Also, longer names can sometime exceed the length of drop down list widths so that only the first several characters can be seen. On the other hand, don't abbreviate so much that someone must rely on the description to figure out what the field contains. Table and field names used to have a limit of 8 characters. Look for the median, where the name is comprehensible, yet of reasonable length.	Low	water_quality_evaluation_code geomorphic_disturbance_description smptrnid	h2o_quality_eval_code geomorphic_disturb_desc sample_transect_id
Specific names	Always choose names that accurately identify the data to be stored in a field. If the field name is too vague, users may enter data other than what you intended for the field.	High	amphibian_size tree_name	amphibian_length_in tree_species_name or tree_common_name
Single value	Always choose a field to contain a single value. Data entry, validation, and retrieval are more difficult if a	High	full_name	first_name, last_name

Recommended Database Strategies

Rule	Explanation	Importance	Examples of poor design		Examples of better design	
	single field contains multiple values.		city_state_zip		city_name, state_code, zip_code	
Avoid calculations	Always choose a field to be independent of all other field values. It is better practice to perform calculations dynamically during a query or report rather than storing the value in the database. Any stored calculation runs the risk of not being updated when one of its elements is updated.	High	line_item_cost		unit_cost, item_qty	
Avoid reserved words	Each database server has a set of reserved words that cannot be used as table or field names. You will not encounter this issue if the recommended naming convention is used, because each field has multiple parts. Still, these reserved words are useful to know. Access may let you create a field with the same name as a reserved word, but this WILL cause problems at a later time, when you create a data entry form or report or try to run queries to view the data. Listed to the right are some of the more common reserved words. See the Access online help system for a complete list.	High	area avg by constraint count counter currency date datetime delete desc exists from group	ignore in index insert key max memo min money name number option order percent	perimeter primary procedure property references report section select set shape sum table text time	type update value values update year yesno

Step 2: Select the data type and category of the field

Field Data Types

Generic Data Type	Access Data Types	Description
boolean	yes/no	Stores values representing yes or no (on/off, true/false).
alphanumeric	text memo	Stores any combinations of letters, numbers, and special characters and should be defined to the maximum allowable length.
numeric	integer long integer	Stores numbers only; both whole numbers and real numbers may be permitted depending on the specific data type.

Recommended Database Strategies

Generic Data Type	Access Data Types	Description
	currency single double	
date/time	date/time	Stores any valid date, time, or datetime.
counter	auto-number	The system generates a unique identifier for the field. Some databases servers allow the designer to choose between sequential or random numbers.
other	oleobject	Stores "blob" data (binary long object), such as images and documents.

Note that declaring a numeric field in Access is a two-part process. From the data type drop down list, select Number. Under the General attributes tab at the bottom of the window, select the specific numeric data type from the drop down for Field Size. This text box is also used to define the length of text fields.

The use of field name suffixes has been adopted in many organizations where data is shared and accessed among a large number of users. By categorizing fields in a standard fashion, we are able to include an additional level of description into the field name. Often, the data type of the field is detectable from the suffix.

Field Type Suffixes

Ending	Explanation	Data Types	Examples of field names	Examples of data in fields
id	ID is reserved for the unique identifier or primary key of a data table. Generally, this is a non-meaningful, sequential field. In Access, the auto-number data type can be used, where the system automatically assigns the next whole number in the sequence. The user may also choose to assign the sequence manually. If the unique identifier contains text, the value should be fixed width. This means that all text must have the same number of characters and the numbers must be padded with zeroes so that each value takes on the same format.	auto-number long text	bird_detection_id tree_id site_id observation_id	1, 2, 3, 25, 300, 90548 534, 2544, 3221 1620101, 1620102, 1630101 ROCK_CR01, ROCK_CR13 20010706, 20010724
num	NUM is reserved for fields containing numbers (and sometimes characters), but for which no calculations will be performed. This field should be assigned to a text data type, even if the field contains all numbers. The data should not be formatted with special	text	gps_unit_num telephone_num zip_code_num	0001, 0002, 0088, 0230 9072993500, 9072221010 99555, 99508, 99501

Recommended Database Strategies

Ending	Explanation	Data Types	Examples of field names	Examples of data in fields
	characters, but may be padded with zeros. Generally, all field entries have the same length (fixed width).		trail_num	DT005, DT062, DT455
name, type	NAME or TYPE is reserved for text fields. This field may contain numbers. It will not usually have a fixed width.	text	water_body_name observer_name gps_type	Pillin Lake, Marlo Creek DAS, Bill Johnson PLGR, Garmond III+
desc, notes, comments	DESC, NOTES, or COMMENTS is reserved for text fields containing formatted text descriptions or comments. Text fields in Access are limited to 255 characters. If a longer length is desired, the memo data type should be selected, which can hold up to 64000 characters.	text memo	site_desc sample_desc observation_comments	{paragraph style text}
code	CODE is reserved for the unique identifier or primary key of a lookup table. This name should be used for the associated field in the data table. This is explained in greater detail below.	text	park_code	ANIA, DENA, KATM, WRST
date	DATE is reserved for fields containing dates with or without the time. If a date is entered without a time, the time defaults to 12:00:00 AM. Dates may be entered in a variety of recognizable formats and may be formatted to display as desired.	datetime	sample_event_date recording_date	06/01/2000, 08/15/2001 09/05/2000 10:34:29 AM
time	TIME is reserved for fields containing times without dates. Since Access stores this as a datetime and no date is provided, the date defaults to 12-30-1899. Times may be entered in a variety of recognizable formats and may be formatted to display as desired.	datetime	start_time	08:22:03 AM, 14:55:22
amt, cost	AMT or COST is reserved for fields containing monetary values. Access has a special currency data type for dollar amounts. This is a numeric data type, so special characters will not be accepted. Dollar signs may be added in the display format.	currency	purchase_cost	95.99, 5020.00, 8333.25
count, qty	COUNT or QTY is reserved for fields answering the question "how many?". Generally, this field will have	integer	bird_count	10, 25, 530

Recommended Database Strategies

Ending	Explanation	Data Types	Examples of field names	Examples of data in fields
	a data type that only allows whole numbers. Integers can hold a numeric value up to 32767. If the count can be greater than that, use the long data type. Less commonly, the count may be able to contain fractions. If so, use the single or double data type.	long	quantity	
percent	PERCENT is reserved for fields containing percentages. This field is usually stored as numerics between 0 and 100, but may also be stored as a decimal between 0 and 1. Data should be stored consistently within a single field. This is a numeric data type, so special characters will not be accepted. Percent signs may be added in the display format.	any numeric	litter_cover_percent cloud_cover_percent	20, 55, 62.5, 80 .2, .55, .625, .8
height, width, length, girth, depth, weight, temp	These suffixes and others are reserved for measurements. These fields can be stored as any numeric data type and calculations may be performed. Unit of measure should also be included, as described in the following section. This may require some conversions prior to data entry, but will help ensure that all measurements are entered in a common unit, rather than mixed. Anyone viewing the database may easily detect the unit, as well.	any numeric	amphibian_length_cm tree_height_ft river_width_ft trail_length_km sample_weight_g	10, 22.5 3, 45 10.7, 150 2.2, 225.8 .85, 5.333
is, are, has (use as prefix, not suffix)	IS or ARE or HAS are prefixes reserved for boolean values, or the answer to yes/no questions. Internally, Access stores -1 for true and 0 for false. This field may be formatted in Access as yes/no, true/false, or on/off.	yes/no boolean	are_deformities_present is_snowing has_litter_cover_gt_20	yes, no -1, 0 true, false

Step 3: Select the unit of measure of the field, if one exists

For fields that require a unit of measure, such as any measurement, it is important to include the unit on the field form as well as in the database field name. It is much easier to ask for a given unit up front, than have to convert all the mixed units into a common one after the data has been collected. Even more difficult is to interpret those numbers where no unit is given, as it is not always clear. Having the unit as part of the field name helps clarify the unit for the data entry person or anyone viewing that data at a later date.

Recommended Database Strategies

Field Unit of Measure Suffixes

Ending	Explanation	Data Types	Examples of field_names
in, ft, cm, m, km	These suffixes and others are reserved to describe the unit of measure for distances. Use the common English abbreviation for any additional units that are utilized.	any numeric	river_depth_ft elevation_m
g, kg, lb	These suffixes and others are reserved to describe the unit of measure for weights. Use the common English abbreviation.	any numeric	amphibian_weight_lb
deg	DEG is reserved for fields measured in degrees. This includes radial degrees 0-360. Decimals may be used.	any numeric	slope_deg aspect_deg
degf, degc	DEGF or DEGC is reserved for fields measuring temperatures. Use DEGF if the field is to be recorded in degrees Fahrenheit and DEGC for degrees Celsius.	any numeric	sample_temp_degc river_temp_degf
mph, cfs	Additional units of measure may be used as needed.	any numeric	animal_speed_mph river_volume_cfs

Step 4: Write a description for the field

Write a clear field description that accurately defines the field.

Step 5: Define other characteristics of a field

Define a Field Validation Rule

Features of Microsoft Access allow you to easily manipulate your data, and more importantly, ensure that the data being entered or modified is valid. In Access, you can create validation rules and setup valid lists of values for use in drop down list boxes (or picks lists). Pick lists give the user a list of options and the user can only choose from that list. This prevents bad data from being entered and also helps to prevent "typos".

Example:

A validation rule might state that the longitude must fall between -149 and -157. Any value entered that falls outside of this range is rejected by the system and triggers an error message. This prevents faulty data such as -14.9088, as well as typing mistakes, such as someone that omits the negative sign and enters 150.8665.

A valid value list might contain the following values: {prehistoric, historic, modern}. Only one of these three options can be selected for the field. A

Recommended Database Strategies

user could not enter *futuristic* because it is not a valid value, nor could a user mistakenly type prehistoric as *p-h-i-s-t-o-r-i-c*. This saves time at a later point when someone has to research this value to decide if the true value should be prehistoric or historic.

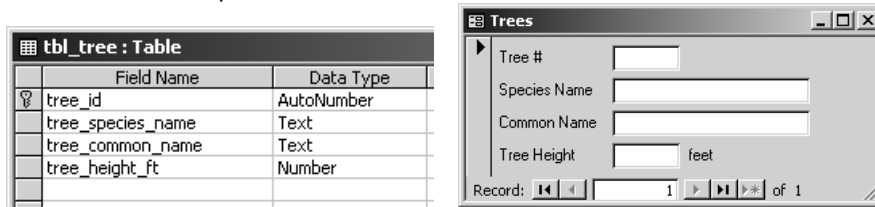
For more details, refer to topics [Restrict or validate data](#) or [Work with lookup fields](#) in the Microsoft Access Help file.

Define a Field Label

Realize the difference between a field name and a field label. The field name is the internal name that is used by the database to track a field. It is the field names you see when viewing an Access table in design mode. The label is the display name for the field, usually placed next to the field text box on a form or report. There are no rules or standards for labels. You may name them in any way you like. My only suggestion is that the same field should have the same label if it appears on multiple forms or reports, so as to not confuse the users into thinking they are actually different fields.

Example:

Observe the differences between the field names on the left and the field labels on the right. I adhere to the naming convention rules for the fields, but the labels can be named as you see most fitting for the users. My labels include the use of spaces, special characters, and mixed-case text. Since I choose "Tree #" to label my tree_id field, I should maintain that same label anytime I use the tree_id field on other forms and reports. I should not use "Tree #" here, then label the same field "Tree ID" or "Tree Num" somewhere else.



The image shows two side-by-side screenshots from Microsoft Access. The left screenshot displays a table named 'tbl_tree : Table' in design view. It has four fields: 'tree_id' (AutoNumber), 'tree_species_name' (Text), 'tree_common_name' (Text), and 'tree_height_ft' (Number). The right screenshot shows a form titled 'Trees' with four input fields labeled 'Tree #', 'Species Name', 'Common Name', and 'Tree Height' (with a 'feet' unit label). The 'Record' bar at the bottom indicates '1 of 1'.

Field Name	Data Type
tree_id	AutoNumber
tree_species_name	Text
tree_common_name	Text
tree_height_ft	Number

For more details, refer to topic [Set field properties to customize how data is stored, handled, or displayed](#) in the Microsoft Access Help file.

Define a Field Format/Input Mask

Often we want to view data formatted in a particular way. Some good examples include dates, telephone numbers, currency amounts, and percents. Do not store the formatted text in the database as a text field. Only the raw data should be stored in the table. Otherwise, you lose the benefits of date and numeric field types, mainly the calculations that can be performed on them. A date, which is entered and stored as "1/1/2" can be formatted however you like, including Julian dates. However, if you convert what the user entered to "January 1, 2002" and save that as a text field instead of a date field, you can no longer compare it with other dates or perform any date calculations. The same goes with currency and percents. Do not store dollar amounts as "\$500.00" in a text field. Store it as "500" in a numeric field, then display it on forms as "\$500.00" using the "Currency" format.

Phone numbers are a little different, because they should be stored in text fields. However, you should still resist the temptation to store the formatting. Phone number fields should be ten characters in length, so the raw data stored in the field might look like "9075551212". To keep users from entering formatting on data entry, you can create an input mask. This assists the users by displaying the formatting while they enter and view data, but the system does not keep that formatting in the database.

Recommended Database Strategies

Example:

Observe the field data types and then the differences between the data stored in the fields and the different ways in which you can format them on forms. The phone number field is displaying the input mask. As mentioned above, this guides the user to enter the correct information. Input masks work with social security numbers and any other field that has a standard format to it.

tbl_miscellaneous : Table	
Field Name	Data Type
birth_date	Date/Time
dollar_amount	Number
contrib_percent	Number
phone_num	Text

tbl_miscellaneous : Table			
birth_date	dollar_amount	contrib_percent	phone_num
6/5/1965	500.2	0.875	9075551212

Miscellaneous

Birthdate: June 05, 1965

Dollar Amount: \$500.20

Contribution: 87.50%

Phone Number: ([###]) ### ####

Record: 1

For more details, refer to topics [Should I use a data display format or an input mask?](#), [Define the data display format for a field in table design view](#), or [Define an input mask for a field in a table](#) in the Microsoft Access Help file.

Other Data Design Topics

Actual Measurement vs. Ranges and Categories

One decision to make is the choice between allowing the user to openly enter any value in a field versus forcing the user to select from a limited list of values. There are valid situations to use each. If a field has a finite set of values, then it is generally best to create a list of these values in a lookup, or reference, table. The main advantage of using a reference table is to prevent someone from typing in a value that is not appropriate for the field, including typing mistakes. An example of this might be a field for the park abbreviation. We know there are a set number of parks and we don't want the user to enter an abbreviation that is not truly a park. If a new park is added, a new value can be added to the list. Lists also prevent incorrect typing so that you don't have multiple spellings of what should be the same name, such as species_name or other scientific names.

While reference tables are generally a better choice for text fields with a restricted list of values, numeric fields are often better left open for user entered values. Two things to consider are: 1) how exact of a value can the data collector detect and 2) how will this field be used/what calculations or summaries will be performed on this field after all the data is collected. An example of this is a field for the percentage of cloud coverage, named cloud_cover_percent. You could leave the field open, in which the user would enter a value between 0 and 100. Or you could create a reference table containing a list of the following ranges: <25%, 25%-50%, and >50%. If you leave the field open, the user must make a best guess at the coverage (which may not be 100% accurate), but you later have exact values that can be used in calculations and summaries. You can group by 10's or you could group in the categories listed or you could create a legend that was color indexed by the exact values. If you have the user select one of the three categories, then you can not group the cloud coverage by anything other than the choices from your list. A compromise in this situation might be to have the data collector give the coverage rounded to the nearest 10%. It would be better to have the values 10, 20, 30...90, 100 stored in the database rather than 1 (which = <25%), 2 (which = 25%-50%), and 3 (which = >50%), which represent ranges. Someone reviewing data in the table doesn't need to know the translation to the lookup values to understand what they are looking at.

Recommended Database Strategies

Data Verification

Manual effort is usually required to get data into an electronic format. Any errors made during typing will be stored in the permanent database unless the data is verified and the errors are detected. By implementing a data verification practice, these data entry errors can be reduced, if not eliminated. There are many methods of verification; with those eliminating the most errors being the most time consuming and vice-versa. Here are some recommended techniques.

Data Verification Method	Advantages/Disadvantages
The data entry person visually verifies each record after it is input. The values recorded in the database are compared with the original values from the hard copy and any errors are corrected immediately.	This method is the least complicated since no additional personnel or software is required. The reliability depends on the person keying data and is generally the least reliable method of data verification.
All records are printed upon the completion of data entry. The values on the printout are compared with the original values from the hard copy. Errors are marked and corrected in a timely manner.	The reliability of this method increases if someone other than the person keying data performs the review. Again, no special software is required.
The data entry person completes all data input, as normal. Random records are selected (every n th record) and entered into an empty replica of the permanent database, preferably by someone other than the person keying the permanent data. A query is run to automatically compare the duplicate records from the two datasets and report on any mismatches of data. These disparities are manually reviewed and corrected, if necessary.	This method involves the overhead of retyping the selected records, as well as the creation of a comparison query (which required additional effort, but is not time-consuming). This method becomes increasingly successful as the value of n decreases.

References

Database Design for Mere Mortals, Hernandez, Michael J., Addison-Wesley, Copyright 1997